Accuracy of distributed timestamps

Celine Valot

UNITÉ DE RECHERCHE
NRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

1 9 9 2

25ème anniversaire

N° 1804

Programme 1
Architectures parallèles, Bases de données,
Réseaux et Systèmes distribués

ACCURACY OF DISTRIBUTED TIMESTAMPS

Céline VALOT

Décembre 1992

Accuracy of Distributed Timestamps

Précision des estampilles réparties

Rapport de Recherche INRIA n^o1804

Céline Valot

INRIA
B.P. 105, 78153 Le Chesnay Cédex, France
tel.: +33 (1) 39-63-59-00, telex: 697 033 F
email: valot@corto.inria.fr

Novembre 1992

Contents

Abstract

Various timestamping mechanisms had been provided in order to characterize the partial order between events occurring in a distributed computation. Using the order induced by such timestamping mechanisms, one can reason about causal or concurrent relationships between events. However comparisons between timestamps can induce false assumptions on these relationships. The order induced by the timestamps is not always the existing partial one in a distributed execution. It is in these terms that we study the accuracy of three timestamping mechanisms, showing the type of compromise (space-accuracy tradeoffs) one has to make when using them.

Résumé

Divers mécanismes d'estampillage ont été produits dans le but de caractériser l'ordre partiel entre les événements d'une exécution répartie. En utilisant l'ordre généré par ces estampilles, il est possible de raisonner sur les relations de causalité ou de concurrence entre les événements. Cependant, les comparaisons entre les estampilles peuvent induire en erreur l'observateur de l'exécution. En effet, l'ordre induit par les estampilles ne reflète pas toujours celui qui existe au sein de l'exécution répartie. C'est en ces termes que nous étudions la précision de trois mécanismes d'estampillage et que nous montrons le type de compromis à effectuer (espace/précision) dans l'utilisation de ces mécanismes.

1 Introduction

Distributed computing commonly refers to an activity performed in a spatially distributed system. One main characteristic of distributed computations is the lack of a common reference to time. And physical clocks cannot be used without specific and expensive algorithms to synchronize them.

However, perception of time is fundamental in comprehending distributed systems, in order to relate causes and effects. Distributed algorithms require the ability to determine wether one of two events comes before or after the other. Events occurring on a single processor are totally ordered, and their ordering can be determined by reading the processor physical clock. Events occurring at different processes are just partially ordered, by processes exchanging messages. A sending of a message always precedes its receiving. A timestamping mechanism is a mechanism which associate to each event of the computation a date, say $date(e)$ to whatever event e, and one would like an exact mapping between the timestamps ordering and the corresponding events occurrence one. Various timestamping mechanisms were proposed to achieve this requirement. Nevertheless, they differ in cost as some of them are space consuming and we show that they differ in accuracy also, difference we are able to quantify precisely.

The timestamping mechanisms we study are the logical clock one from Lamport, the vector time one from Mattern and the interval clock from Jard. Those three mechanisms run obeying to nearly the same rules, i-e incrementing the timestamps upon occurrence of an event, and updating the timestamp value on receiving a message. However those timestamps have different sizes, leading to different results when characterizing causality. Our motivations are that, everyone agree to say that Lamport's mechanism is imprecise, whereas Mattern's one is precise but expensive but their accuracy degree has never been studied. The known gap between those two mechanisms has never been measured. By analysing their different behavior, we propose an accuracy measure of these timestamps, and evaluate the complexity of this measure. We identify the compromise to make, concerning the *accuracy* obtained relative to the space occupied by the timestamps.

For that purpose, we show in the first part of this document (Section 2 and Section 3) that timestamping mechanisms produce an order between events which is different from the one effectively realized. By examining the points of these timestamp orders, we approximate the amount of errors that will be made by the timestamps. Using a different approach in the second part of this document (Section 4), we show that the same results are obtained adopting a differential calculus of the accuracy measure. Section 6 concludes by showing the space/accuracy compromise

one has to make when using timestamps.

2 The timestamping mechanisms studied

2.1 The model used

The distributed computing model used in this paper is a classical model in which main entities are *processes*. Sequential processes execute concurrently in order to accomplish a given task. They interact by communicating by message passing in an *asynchronous* manner.

A distributed execution is the execution of a possible behavior on a group of processes, possibly located on different sites. An execution is composed by events, those entities that are timestamped. Three types of events are considered, namely send events, receive events and internal ones. A *send event* is the sending of a message by a process to another one, a *receive event* is the receipt of a message by some process, and finally, an *internal event* is one that has no influence on the rest of the system.

Notations and definitions

The notations given here are from [Diehl 1992].

For each event e, we note by e^- its predecessor on the same process, by e^+ its successor also on the same process, by e^r the corresponding receiving event if e is a sending of a message, and by e^e the corresponding sending event if e is a message receipt.

The causality relation (also called "happened-before") is a relation between any two events, noted :

$$\theta(e, f) \iff e \; causally \; precedes \; f.$$

The causality relation is a transitive relation.

If neither e causally precedes f, nor f causally precedes e, they are said to be *concurrent* events, which is noted $(e \parallel f)$. The concurrency relation is a symetric relation but not transitive.

The set of events E of a distributed computation with the causality relation forms a partially ordered set, or poset for short, and is noted (E, θ).

The set of predecessors of an element $x \in E$ is defined as :

$$Pred(x) = \{y \in E \ / \theta(y, x)\}$$

A *chain* is a subset of the overall set of events E of pairwise comparable elements. An *antichain* is a subset of E containing pairwise incomparable elements, i-e concurrent.

We recall here briefly the timestamping mechanisms of Lamport, Mattern and Jard [Lamport 1978, Schwarz 1991, Diehl 1991].

2.2 Logical clocks

Lamport introduced *logical clocks* (LC) to timestamp events of a distributed computation. Logical clocks are integers incremented upon an event occurrence, and which conform to the causality relation :

$$\text{if } \theta(e, f) \Longrightarrow LC(e) < LC(f)$$

However, it has been revealed that logical clocks can't fully characterize causality. Indeed, concurrent events can either get the same timestamp, or can obtain different ones (see for example Figure 2, where LCs are illustrated by a single integer). One would like that two equal timestamps should mean that the corresponding events are concurrent. By solely comparing the timestamps of two events, one cannot determine if they are causally related or not. It is well known that the logical clock ordering is a linear extension of the underlying partial order, producing then a total order. This total ordering yields to a misunderstanding of the events ordering.

2.3 Vector time

Mattern used vectors of logical clocks, also called *vector time*, whose size is equal to the computation processes number. Vector time has the property of completely characterizing causality in the sense that it is sufficient to compare two vectors to determine if the two corresponding events are related or not.

Each process P_i is given a vector V_i of n (n being the number of processes) clocks and increments its i^{th} ($V_i[i]$) element upon occurrence of every event. On sending a message, P_i piggybacks its current (incremented) vector on that message. The receiving process P_j increments its j^{th} ($V_j[j]$) element and then updates its knowledge of every processes vector time, by performing $V_j = max(V_j, V_t)$. V_t being the

received vector (see Figure 2 where VCs is an array of three integers). Interestingly, $\forall\ P_i\ and\ P_j, V_i[i] \geq V_j[i]$[1], and $V_j[i]$ is an approximation held by P_j of P_i's i^{th} vector time component. Vector time has the following properties :

$$\forall\ e, f \in E,\ e \in P_i,\ f \in P_j$$
$$(i)\ \theta(e,f) \text{ iff } V_i(e)[j] < V_j(f)[j]$$
$$(ii)\ e \parallel f \text{ iff } V(e) \parallel V(f) \begin{cases} V(f)[i] < V(e)[i] \\ and \\ V(e)[j] < V(f)[j] \end{cases}$$

Without knowing nothing about the underlying computation, vector time appears to be the timestamping mechanism exactly characterizing causality.

Due to the large size of vector time, optimizations were proposed in order to reduce it. It appears that the reduced vectors fail to fully characterize causality. B. Charron-Bost proved that characterizing causality in a system of M processes requires vectors of at least size M, if nothing is known about the computation [Charron-Bost 1991]. The proof rests on the fact that a computation of M processes induces a partially ordered set $(E, \rightarrow)$ of *dimension* M[2] and that this set can be embedded in a partially ordered set $(N^k, <)$ of integer values. For characterizing causality, it is required that $k \geq dim(E, \rightarrow)$.

To summarize, there exists an isomorphism between vector time ordering and the events ordering, but vector time have the drawback of being space consuming.

2.4 Interval clocks

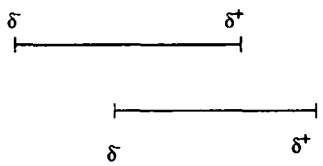

Figure 1: Overlapping intervals

Jard in [Diehl 1991] proposed a new stamp mechanism based on the class of interval orders (see definition 4, section 4.3). For each event e of the computation, an interval

[1] P_i has the most accurate knowledge of its own virtual time.

[2] The dimension of a partially ordered set $(A, <)$ is the cardinal of the minimal collection of linear extensions of $(A, <)$ such that their intersection is equal to $(A, <)$.

clock is associated, of the form $[\delta(e)^-, \delta(e)^+[$. They are computed in the following manner :

- If e is an internal event :

$$\begin{cases} \delta(e)^- = \delta(e^-)^- + 1 \\ \delta(e)^+ = \delta(e^+)^- \end{cases}$$

- If e is the sending of a message :

$$\begin{cases} \delta(e)^- = \delta(e^-)^- + 1 \\ \delta(e)^+ = \min(\delta(e^+)^-, \delta(e^r)^-) \end{cases}$$

- If e is the receipt of a message :

$$\begin{cases} \delta(e)^- = \delta(e^e)^- + 1 \; if \; \theta(e^-, e^e) \; or \; \theta(e^e, e^-) \\ \delta(e)^- = \max(\delta(e^e)^-, \delta(e^-)^-) + 2 \; if \; e^- \; || \; e^e \\ \delta(e)^+ = \delta(e^+)^- \end{cases}$$

Those intervals produce an interval extension of the partial order; they are somewhat more precise than logical clocks as an overlapping of two intervals means that the associated events are concurrent (see Figure 1). Interval clocks do not need so much space as vector time does, but they remain however imprecise relative to Mattern's mechanism.

The Figure 2 illustrates the three timestamping mechanisms described above.

3 The timestamps accuracy

3.1 Definitions

We want here to provide definition of the word *accuracy* of timestamping mechanisms.

The accuracy of a timestamping mechanism is the amount of errors made in the answering of the following question : given any event e and f, and looking at their timestamps, does e causally precedes f ? This question can be asked analogously for concurrency relationships, i-e events that are not related [3].

[3] We speak of concurrent relationships which can be surprising at a first glance, as concurrent events are not related, but in our text this absence of relation forms a relation which we call *concurrent relation*.

The events e and f are either causally related in the execution or are concurrent. We want to measure in which way we can trust comparisons between timestamps in order to estimate the precision of our knowledge about the actual ordering of the execution events. It has been shown that the order produced by the timestamps, either creates relations which are inexistent in the underlying computation, or suppress existing ones. Obviously, the better accuracy will be one of a timestamping mechanism whose order is in exact correspondence with the underlying partial order.

Definition 1 *The* **accuracy** *of a timestamping mechanism is the ratio of the number of causal or concurrent relationships correctly induced by the timestamps over the number of existing relationships (causal or concurrent) between each pair of the overall set of events.*

As it is difficult to provide fully general results concerning accuracy of timestamping mechanisms, we will give for each of them a worst and best case.

In [Diehl 1992], a tentative of measuring the complexity of timestamping mechanism was done considering three quantities :

- the amount of memory storage for the timestamps;
- the cost of calculating the timestamps;
- and the cost of the answer to $e < f$?

Those quantities are indeed interesting but in addition to the cost of answering to the question $e < f$, we are interested by the correctness of this answer. A wrong answer could have serious consequences.

We determine here the accuracy of the timestamps of Lamport, Mattern and Jard, that is we aim at approximating the amount of errors made in the identification of the causality/concurrency relationships.

Whatever the computation being observed, by order of its structure, Mattern's vector time is known to be the only mechanism characterizing causality and concurrency. It is however a great consumer of memory space. We study here the accuracy of the two other timestamping mechanisms relative to vector time. This allows us determine the compromise to make when using them.

Proposition 1 *The vector time mechanism accuracy, noted $\mathcal{A}(\mathcal{V})$, is equal to 1.*

If we mean by accuracy the number of correct causal or concurrent relationships identified by the timestamping mechanism, $\mathcal{A}(\mathcal{V})$ is equal to one. If we call ρ the total number of relationships in the execution, whose value is some m, and by ϱ the number of such relations infered by vector time, it appears that its value is also m. Thus the ratio ϱ/ρ equals 1. This result shows that vector time characterizes causality and concurrency in an exact manner.

Proposition 2 *The product of accuracy measure with the space occupied is a constant.*

Vector time accuracy multiplied by the space occupied yields to a constant, $1 \times n = n$.

3.2 Accuracy of Lamport clocks

The Figures 3 and 4 are examples where, concurrency is maximal and causality is complete, respectively. In the first case, there are a lot of antichains, and in the second case, the overall set of events forms a chain.

The behavior of Lamport logical clocks differs in an important way from these two examples. We note that in case of maximal concurrency, the clocks fail to identify concurrency relationships (a lot of concurrent events are ordered by their timestamps).

However, for the complete causality example, logical clocks conform precisely to the causal ordering and have an accuracy of 1. Intuitively, in presence of antichains, the accuracy of logical clocks could approximate some value n. Those two examples are respectively the worst (maximal concurrency) and best (complete causality) cases. We will detail these in Section 3.3.

We are able to obtain in an exact manner the error number made by the logical clocks, fact that we show below. Those errors come from the presence of antichains, but vary with the event sequence.

In Charron-Bost PHD thesis, measures of a distributed computation parallelism are introduced : the w measure takes into account antichains of length 2, that is numbering pair of concurrent events; the m measure counts all the antichains, leading to the numbering of all consistent cuts[4].

[4]

Definition 2 *$\forall c \in C, C \subseteq E$, $\forall e' \in E$ such that $e' \rightarrow c$, C is a consistent cut iff $e' \in C$.*

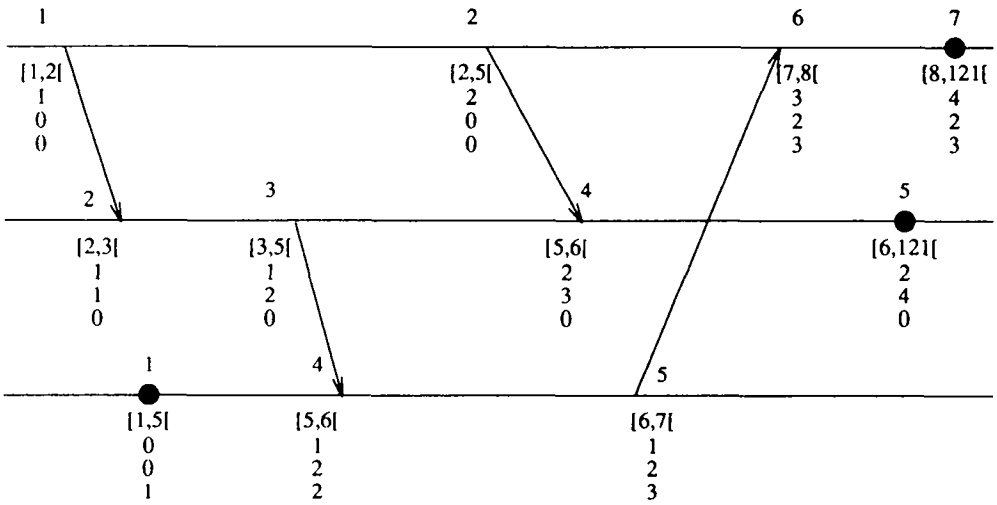

Figure 2: Timestamping mechanisms

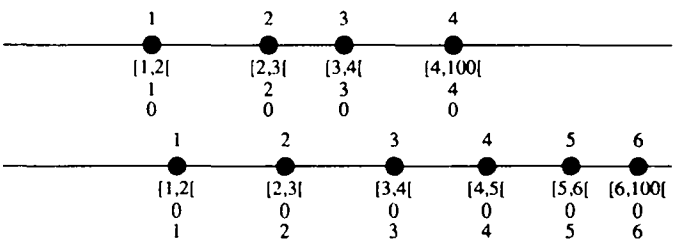

Figure 3: Maximal concurrency

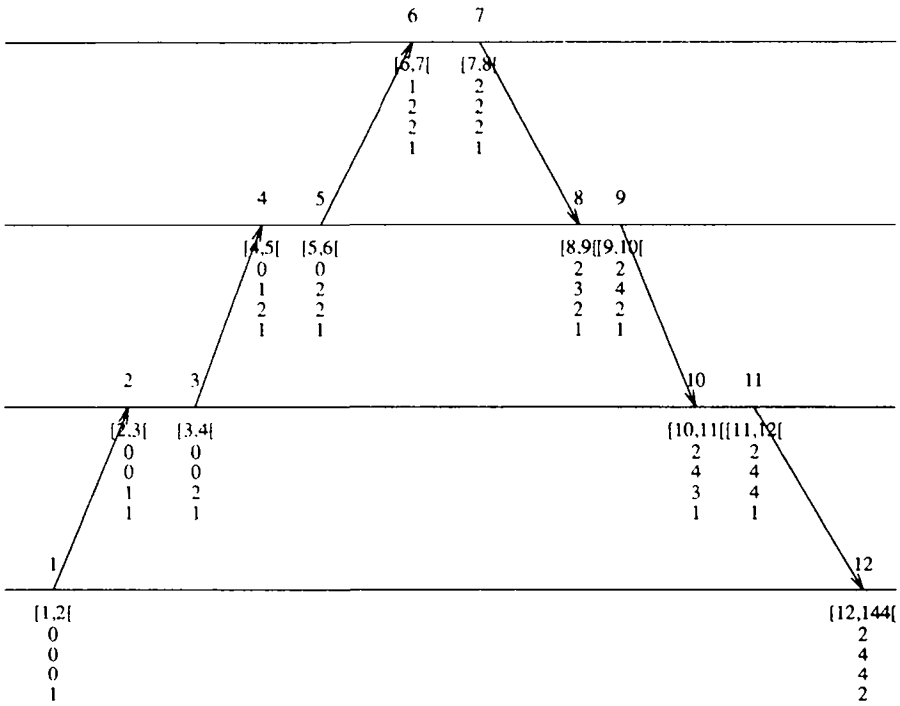

Figure 4: Complete causality

The maximal number of antichains of length 2 is given by the following formula :

$$\sum_{1 \leq i < j \leq n} q_i q_j$$

with q_i, q_j being the total number of events on processes P_i and P_j. The Figure 3 shows an example of concurrency where this bound is reached.

From that result, it is quite natural to determine the number of pair events pertaining to antichains that will not be detected to be concurrent by the *LC*s. Hence, our interest resides in antichains of length two.

Definition 3 *Let $\mathcal{E} = (E, \theta)$ a partially ordered set. The rank of an element x is the length of a maximal chain ending in x.*

The set

$$\mathcal{R} = \biguplus_{0 \leq i} R_i \text{ with } \forall\, i \geq 0, R_i = \{x \in E, r(x) = i\}$$

is called a rank decomposition of $\mathcal{E}$. Elements having the same rank are non-ordered.

Lamport clocks correspond to an antichain decomposition of the causality relation [Diehl 1991]. Using the concept of *rank* (see definition 3 above), events x and y are showed to be concurrent if they have the same rank, $r(x) = r(y)$. One can note that the LC's characterize perfectly concurrency between elements of the same rank. The proof is straightforward.

Proof:

Let's take x and $y \in E$ such that $r(x) = r(y)$. That means that there exists a chain L_x ending in x, x being maximal, and a chain L_y ending in y, y being also maximal. Assume then that $\theta(x, y)$, then $x \in L_y$ and x is not maximal. Or assume then that $\theta(y, x)$ then $y \in L_x$ then y is not maximal. This contradicts the definition and shows that if $r(x) = r(y) \Leftrightarrow x||y$.

□

This proves that elements of the same rank are concurrent. Let's show now that two events having the same rank get the same timestamps. The proof is made by induction on the number of events pertaining to a chain.

Proof:

Let's take x and y such that they are the first events on their respective chains. These chains are obviously distinct. The L_x chain ends in x then $LC(x) = 1$. This holds also for y in L_y and we have $LC(x) = LC(y)$. If x is not minimal, we suppose that the previous arguments are true for all predecessors of x. That is $LC(x^-) = LC(x_0) + (n-1)$. As $LC(x) = LC(x^-) + 1$, $LC(x)$ is also equal to $LC(x_0) + (n-1) + 1$. And this is true in the same manner for y, which shows that $LC(x) = LC(y) = LC(x_0) + n = LC(y_0) + n$ for induction.

□

This shows that events having the same rank will get the same timestamps.

However there exist concurrent events which do not have the same rank. And this feature is not detected by Lamport clocks. We show hereafter the problems arising.

Given $x \parallel y$. x is concurrent with all the elements which y is in relation with, but with some restrictions enounced below. This is actually not detected by logical clocks.

We want to know the error number concerning (given $x \parallel y$) the relationships of y with $(\restriction x) = \{z \in E / \theta(x, z)\}$, the *future* of x, as of y with $(\downharpoonright x) = \{z \in E / \theta(z, x)\}$, the *past* of x. This is done in a similar way for x with the elements pertaining to the chain to which y belongs.

The number of relations that won't be identified by the logical clocks is given by the following formula. For some x :

$$|(\downarrow y)| - |(\downarrow y) \cap (\downarrow x)| + |(\uparrow y)| - |(\uparrow y) \cap (\uparrow x)|$$

where $(\downarrow x)$ denotes the past of x and $(\uparrow x)$ its future. Symmetrically, the same formula holds for y. This means that x will be concurrent with the cardinal of the past of y minus the intersection of the pasts of x and y, and also with the cardinal of the future of y and with the same restrictions on their intersecting futures. If the intersection set $(\downarrow y) \cap (\downarrow x)$ is empty, and the same for the future, we are in the worst case.

These remarks lead us to identify the number of errors that will be made by the LC's.

Let's note $e_1, e_2, \cdots, e_n$ the events set belonging to $\bigcup_{1 \leq i \leq n}^{n} R_i$. The number of concurrency relationships ($\mathcal{I}$) identified by the LC's is :

$$\mathcal{I} = \sum_{i=1}^{n} \frac{|R_i||R_i - 1|}{2} \tag{1}$$

The number of errors (non-identified relations) ($\mathcal{J}$), for whatever $x \in L_x$, $y \in L_y$ such that $x \parallel y$ and $\exists\, z \in L_y, t \in L_x$ is :

$$\mathcal{J} = \frac{|L_x||L_y - 1|}{2} - |\{z \in L_y / \theta(x,z), \theta(z,x)\}| - |\{t \in L_x / \theta(t,y), \theta(y,t)\}| \tag{2}$$

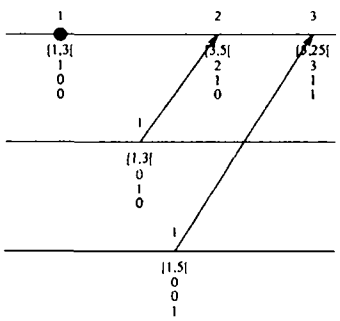

Figure 5: Error number equals to 1

The event $z \in L_y$ in formula 2 appears to be the first successor or the first predecessor in chain L_y (depending if we look at the future or at the past) of event $x \in L_x$ and analogously for $t \in L_x$ for event $y \in L_y$.

The greater is the number of concurrency relations, the greater will be the number of errors. If the computation is made solely of chains, the error number will be small, possibly inexistant (see example of Figure 4). However, this does not allow us to devise a general rule. Taking an example where causality and concurrency are well sub-divided shows that the errors depend on the event sequence. On the Figure 5, there is one error and on the Figure 6, there are three. These two examples have however the same measure w of parallelism, that is the same number of pairs of concurrent events. This remarks shows that we couldn't use this measure of computation parallelism as a measure of timestamping mechanisms accuracy, mainly because its purpose is more general than accuracy matters, and leads to imprecision if applied to our purposes.

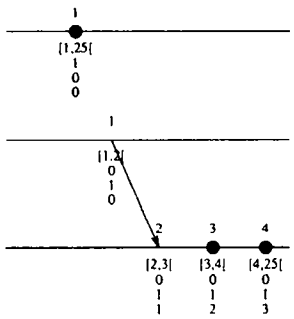

Figure 6: Error number equals to 3

3.3 Best and worst cases

The worst case is a case where concurrency is maximal and the reader can be easily convinced of that, by imaging the existence of a sending/receipt event involving any events of the computation showed in Figure 3. This virtual event creates a relation which matches the one produced by the LC's. This artificial relation can be created at the beginning, the middle or the end of the computation and this choice will have an influence on the length of the chain created. The importance of this chain length will decrease the error number.

Analogously, for the best case where the overall set of events constitutes a chain, it is sufficient to suppress a causal relation by removing a sending/receiving arrow

and see that the IC's mechanism will produce an error. And this error will be propagated to the past or future of the concerned events.

In the average case, it is difficult to identify the number of errors that will be made. It depends on the configuration of the event sequence. It is possible to imagine an execution scenario creating combinatorial relationships between events, with a way to calculate the errors automatically. Maybe, it will allow us to say something on the average case.

This description of the number of identified and non-identified causal relationships constitutes one of our main result.

3.4 Accuracy of interval clocks

Let us see now the accuracy of interval timestamps. Concurrency between events stamped by interval timestamps can be detected if the corresponding intervals do overlap.

$$x \parallel y \Leftrightarrow I_x \; overlaps \; I_y$$

In the worst case, the behavior of interval timestamps is the same as Lamport clocks. Having solely internal events, they are calculated in the same way (see Figure 3). In the best case, interval timestamps conforming to causality, their behavior is also the same (see Figure 4).

In the average case, the reader's has to note that equality between timestamps is not required anymore but the overlapping of intervals. We note however that in presence of specific antichains of length 2, the behavior of interval timestamps will be less erroneous.

In an execution where the immediate predecessor of a receiving event and the corresponding send event are unrelated, the timestamping mechanism will correctly

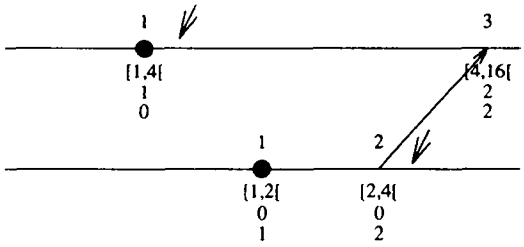

Figure 7: Specific antichains

establish this absence of relations. This is illustrated by Figure 7. In this Figure, the events timestamped by the intervals $[1,4]$ and $[2,4]$ are concurrent, and the intervals overlap. As there is one chance over two that two events be independent, the number of errors is potentially divided by two.

It is sufficient to obtain the number of non-identified relations by interval timestamps to divide by two the number $\mathcal{J}$ given by the formula 2.

3.5 Evaluating the complexity of the accuracy measure

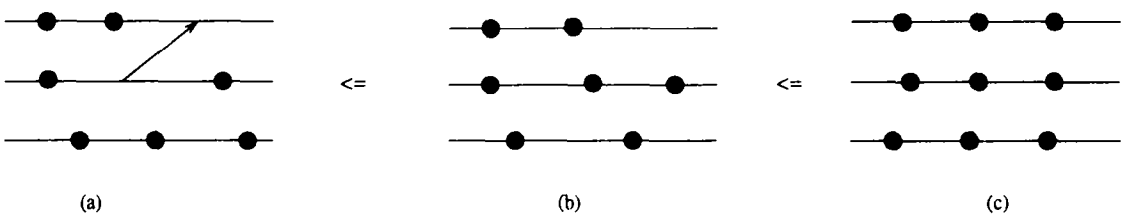

Figure 8: Error number bounds

The Figure 8 illustrates a three-cases example allowing us to estimate bounds on the error number made by the timestamps.

The minimal error number (EN) which can be made is the one corresponding to the best case which we talked about in section 3.3 and which corresponds to the Figure 4. The case (a) of Figure 8 is a case where the EN depends on a chain involving two distinct processes, but which remains greater than the best case number as there remain a lot of concurrent events. The EN corresponding to the case (b) of Figure 8, where concurrency is maximal but with $q_i \neq q_j, q_i < q_j$ (q_i and q_j are the number of events on processes P_i and P_j respectively) is greater than the one of case (a) (This is indicated on the Figure by the sign $<=$). It is itself bounded by the EN of case (c) where concurrency is maximal and $\forall i,j, q_i = q_j$. It is this upper bound that we evaluate below.

The EN can be determined by the following formula:

$$EN = \frac{1}{\sum_{i=1}^{n} q_i}\left(\sum_{i=1}^{n}\sum_{j=i+1}^{n} q_i q_j - \sum_{k=1}^{r}\frac{|R_k||R_k-1|}{2}\right)$$

where r is the maximal rank of the computation and n the number of processes. This means that the value of EN can be obtained by dividing the maximal number of

antichains of length two ($\sum_{i=1}^{n}\sum_{j=i+1}^{n} q_i q_j$) minus the number 1 obtained in section 3.2 ($\sum_{k=1}^{r} \frac{|R_k||R_k-1|}{2}$) by the overall number of events ($\sum_{i=1}^{n} q_i$) on the n processes.

Let's take: $\forall i,j, e = q_i = q_j$ which corresponds to the upper bound of EN as explained above and illustrated by Figure 8.

We obtain the following equations:

$$\begin{aligned} EN &= \frac{1}{ne}\left(\sum_{i=1}^{n}\sum_{j=i+1}^{n} e^2 - \sum_{k=1}^{e}\frac{|R_k||R_k-1|}{2}\right) \\ &\Leftrightarrow \frac{1}{ne}\left(\frac{n(n-1)}{2}e^2 - \frac{n(n-1)e}{2}\right) \\ &\Leftrightarrow \frac{1}{ne}\left(\frac{n(n-1)e(e-1)}{2}\right) \\ &\Leftrightarrow \frac{(n-1)(e-1)}{2} \end{aligned} \tag{3}$$

Proposition 3 *The number of errors EN above can be evaluated in $O(n)$ time complexity where n is the number of processes.*

4 Another approach to accuracy

We present in this section another approach to the problem of calculating the accuracy of distributed timestamps. This approach uses a differential calculus for accuracy estimates. We obtain the same results with this approach as with the one presented in previous sections.

4.1 Creating a generic stamp

We define in this section a generic stamp

$$GC(e) = \sum_{i=1}^{n} VC_i(e)$$

computed for each event e from the corresponding vector time $VC(e)$, by summing the elements of $VC(e)$. The practical use of this stamp is reduced but it will

serve us to establish the accuracy of the different timestamping mechanisms. We will see in section 4.2 that Lamport logical clocks can be expressed using this just defined generic timestamp.

One can note that the sum of the vector components is the exact number of events causally preceding e on all processes.

To show that this generic timestamp constitutes a timestamping mechanism, we have to show that it conforms at least to the causality relation, in the same way as Lamport clocks does.

Proposition 4 *The generic timestamping $GC(e)$ conforms to the causality relation.*

Proof:
Our $GC(e)$ fonction is :

$$GC(e) = \sum_{i=1}^{n} VC_i(e)$$

and we show that :

$$\forall x, y \in E, if \theta(x, y) \Rightarrow GC(x) < GC(y)$$

Let's assume $x \in P_i, y \in P_j$ such that $VC(x) < VC(y)$.

Using the properties of vector time, we can say that :

$$VC(x)[i] \leq VC(y)[i] \quad (4)$$
$$VC(x)[j] < VC(y)[j] \quad (5)$$

Corollary 1 *If x and y are two events of a distributed computation such that $\theta(x, y)$ and if $y \in P_j$, then $VC(x)[j] < VC(y)[j]$.*

The property 5 is the corollary 1 enounced in [Charron-Bost 1989] (recalled above) and allow to affirm that :

$$\sum_{i=1}^{n} VC_i(x) < \sum_{i=1}^{n} VC_i(y) \Rightarrow GC(x) < GC(y)$$

The proof given for this corollary uses the fact that :

$$(\downarrow x)_j = \{t \in C_j, t \preceq x\} \subseteq \{t \in C_j, t \preceq y\} = (\downarrow y)_j.$$

where C_j denotes a consistant cut, $(\downarrow x)_j$ denotes the past of event x, which is equivalent to $Pred(x) \subseteq Pred(y)$. Our $GC(e)$ fonction actually conforms to the causality relation.

□

This generic timestamp does not characterize causality as it is coded with one integer and we know this size is not sufficient to characterize causality.

The purpose of the next sections is to show that we can express Lamport clocks and Jard clocks in terms of this previously defined generic timestamp.

Also we calculate their accuracy using a differential calculus. The use of a differential calculus should rest on the fact that the fonctions are continuous which is not the case. This leads us to use sub-differential calculus instead.

We assume also that the accuracy of vector time is 1, each element of the concerned vector being known at an accuracy of 1. That is, $\forall e$:

$$\begin{aligned} |\partial VC_i(e)| &= 1 \\ |\partial VC(e)| &= |\partial max(VC_i(e))| = 1 \\ |\partial VC(e^r)| &= |\partial max(VC(e^-), VC(e^e)) + 1| \end{aligned} \tag{6}$$

where ∂ denotes the sub-differential operator and not the partial derivative one.

4.2 Accuracy of Lamport clocks

From the above generic timestamp, we can easily create an instance of Lamport clock, called LC_1, by the following equality.

$$LC_1(e) = GC(e) = \sum_{i=1}^{n} VC_i(e)$$

for each event e pertaining to the computation.

Keeping in mind the result 6 enonced above, intuitively the accuracy of this instance of Lamport clocks should be n. This result can be proved showing the linear dependency of the LCs relative to the generic timestamp, and hence, relative to vector time.

Proposition 5 *$\forall e \in E$, there exists a linear dependency between $LC(e)$ and the vector time sum, $\sum_{i=1}^{n} VC_i(e)$, associated to e.*
Therefore $\forall e \in E$, $LC(e) = k \sum_{i=1}^{n} VC_i(e) + l$.

where k and l depend on the events sequence. The expression of LCs in terms of VC's is a piecewise linear fonction. The proof is made by induction on n.

Proof:

Let e_1 being the first event on process P_i, then $LC(e_1) = \sum_{i=1}^{n} VC(e_1)$. And this is true for all the first events on processes $P_1 \cdots P_n$.

Suppose that :

$$LC(e^-) = k \sum_{i=1}^{n} VC(e^-) + l$$

Then e^+ will be written according to two cases :
If e^+ is an internal event or a message sending :

$$VC(e^+) = VC(e^-) + 1$$

and therefore

$$LC(e^+) = k' \sum_{i=1}^{n} VC(e^+) + l'$$

If e^+ is an e^r (a message receipt), its timestamp depends on the maximum of the timestamps of e^- and e^e :

$$\begin{cases} \frac{1}{2}(LC(e^-) + LC(e^e)) + 1 \leq LC(e) < LC(e^-) + LC(e^e) \\ \frac{1}{2}(\sum_i VC_i(e^-) + \sum_i VC_i(e^e)) + 1 \leq \sum_i VC_i(e) < \sum_i VC_i(e^-) + \sum_i VC_i(e^e) \end{cases}$$

by induction hypothesis:

$$\begin{cases} \exists k^-, l^- : \ LC(e^-) = k^- \times \sum_i VC_i(e^-) + l^- \\ \exists k^e, l^e : \ LC(e^e) = k^e \times \sum_i VC_i(e^e) + l^e \end{cases}$$

that is, $\exists k_{min}, k_{max}$:

$$\begin{cases} k_{min} \times \sum_i VC_i(e^-) \leq LC(e^-) < k_{max} \times \sum_i VC_i(e^-) \\ k_{min} \times \sum_i VC_i(e^f) \leq LC(e^e) < k_{max} \times \sum_i VC_i(e^e) \end{cases}$$

by summing these last two inequalities, we deduce :

$$\frac{k_{min}}{2} \sum_i VC_i(e) \leq LC(e) < 2 \times k_{max} \sum_i VC_i(e)$$

Therefore, there exists $k_{min}/2 \leq k < 2 \times k_{max}$ and l such that:

$$LC(e) = k \times \sum_i VC_i(e) + l$$

□

We get for the logical clocks accuracy, noted $\mathcal{A}(\mathcal{L})$, :

$$|\partial VC_i(e)| = 1$$
$$\mathcal{A}(\mathcal{L}) = |\partial LC| = n$$

Again the product accuracy × space is equal to a constant, $(n \times 1 = n)$.

4.3 Accuracy of interval clocks

Using the generic timestamp defined in section 4.1, the interval clocks can be given by the following equation :

$$JC(e) = [GC(e)^-, GC(e)^+[\qquad (7)$$

with $GC(e)^- = \sum_{i=1}^{n} VC_i(e)$ and $GC(e)^+$ calculated as $\delta(e)^+$ (see Section 2.4).

We show hereafter the way our generic interval timestamping mechanism runs.

- If e is an internal event or a receipt of a message[5] :

$$\begin{cases} GC(e)^- = \sum_{i=1}^{n} VC_i(e) \\ GC(e)^+ = GC(e^+)^- \end{cases}$$

- If e is a sending of a message :

$$\begin{cases} GC(e)^- = \sum_{i=1}^{n} VC_i(e) \\ GC(e)^+ = \min(GC(e^+)^-, GC(e^r)^-) \end{cases}$$

where $VC_i(e)$ is the i^{th} vector time element associated to event e. The way $\delta(e)^+$ is computed in case of sending a message ensures the generic timestamp mechanism consistency (the timestamp of a message sending should be less or equal to this message receipt one).

We show hereafter that the generic timestamp defined above is an interval order whose definition is stated below and using the theorem introduced in [Diehl 1991].

Proposition 6 *The interval generic timestamp JC(e) previously defined actually constitutes an interval order.*

Definition 4 *(E, θ) is an interval order if one can associate to each element x of E one interval I_x of the real line, such that :*
$\forall x, y \in E, \theta(x, y) \Leftrightarrow I_x$ is on the left of I_y.

Theorem 1 *[Diehl 1991] The set $([\delta(x)^-, \delta(x)^+[ )_{x \in E}$ where : δ^- satisfies $\forall x, y \in E, \delta(x)^- \leq \delta(y)^- \Longrightarrow Pred_e(x) \subseteq Pred_e(y)$ and $\delta(x)^+ = \min\{\delta(y)^- / \theta_{im}(x, y)\}$ is an interval representation.*

where $\theta_{im}(x, y)$ says thay x is an immediate predecessor of y. An interval representation is an assignment of intervals on the real line to events which satisfies definition 4.

Proof:
Our δ^- fonction is :

[5] Jard's mechanism in case of a message receipt determines if the two predecessors of the receipt event are causally related or not and add +1 or +2 accordingly.

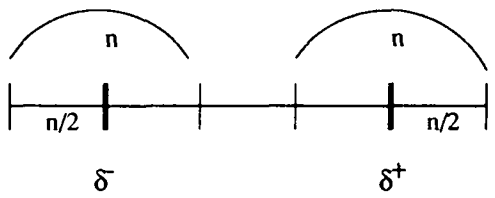

Figure 9: Accuracy of an interval order

$$GC(x)^- = \sum_{i=1}^{n} VC_i(x)$$

and we show that :

> if $\theta(x, y)$ then I_x is on the left of $I_y \Leftrightarrow GC(x)^- < GC(y)^-$
> and $GC(x)^+ \leq GC(y)^-$.

The last proposition is trivial as

$$GC(x)^+ = \min\{GC(y)^- / \theta_{im}(x, y)\} = \min\{GC(y)^- / \theta(x, y)\}.$$

We have to prove that $GC(x)^- < GC(y)^-$, and this part of the proof is exactly the same as the proof of LC_1 mechanism and one can use the proof in section 4.1 on page 16.

This proves that $JC(e) = [GC(e)^-, GC(e)^+[$ fonction actually constitutes an interval order with the same properties as the one defined in [Diehl 1991].

□

We will see in section 5 that a great number of timestamps of this type, i-e coded using an interval order, can be generated in this way.

The accuracy of the interval endpoints $[GC(x)^-, GC(x)^+]$ for whatever x is :

$$\begin{aligned} |\partial GC(x)^-| &= |\sum_{i=1}^{n} \partial VC_i(x)| = n \\ |\partial GC(x)^+| &= n \end{aligned}$$

In fact, the accuracy of $GC(x)^-$ for the clock previously defined (see section 4.3) is at best n, as being a sum of n components whose accuracy is known to be 1. It

is the same for $GC(x)^+$. As event occurrences are independent (in the probabilistic sense[6]), we can therefore conclude that the accuracy of $GC(x)^-$ and $GC(x)^+$, taken individually, is independent. That is the accuracy of the interval clock, noted $\mathcal{A}(\mathcal{I})$ is equal to $n/2$. The Figure 9 illustrates the accuracy of an interval order.

Each endpoint of the interval being known at $+$ or $-n$, the interval has a range of $n/2$, leading an uncertainty about the number of concurrency/causality relations identified. This is due to the information loss resulting of the memorizing of an interval rather than a vector time.

5 Towards a family of timestamping mechanisms

These results on the accuracy of a generic timestamp, expressed as the sum of the elements of the vector time timestamp leads us to propose a generalization of these timestamping mechanisms.

It consists to consider not anymore the sum of components but subsets of elements. As we can choose various such subsets, it yields to obtain a *family of timestamping mechanisms* which could be tailored for each execution, according to their specific characteristics. We have called this family of timestamps *k-vectors*.

Definition 5 *A k-vector is whatever timestamp built by taking a subset of the vector time components. The size of the k-vector is k, corresponding to the cardinal of the subset chosen.*

One first and obvious k-vector is a 1-vector timestamp of size 1, which exactly corresponds to the LC's. Another immediate k-vector is a n-vector timestamp corresponding to VC's. An interval timestamp as defined in section 2.4 is also a 1-vector but which is coded using interval orders. Other k-vectors can be obtained by decomposing the original vector time in k groups of elements. Thus a 2-vector will be a vector time on which we applied a dichotomy on components, $KV_2(e) = (x_1(e), x_2(e))$[7], where $x_1(e)$ is the timestamp of event e for the first group, and $x_2(e)$ the timestamp of the same event e, for the second group. $KV_2(e)$ will have an accuracy of $n/2$. Furthermore using an interval order representation, we obtain this kind of timestamp :

[6] The accuracy of the GC^+ is not subject to the GC^- one.

[7] Parenthesis are used here to avoid confusion with the corresponding 2-vector coded using interval order.

$$JC_2(e) = \begin{bmatrix} [x_1(e)^-, x_1(e)^+] \\ [x_2(e)^-, x_2(e)^+] \end{bmatrix}$$

It have a better accuracy of $n/4$. The reader can easily imagine that if there are $k = n/p$ sets, each of p components, the accuracy of each k-vector will be $p/2$. We conclude that the accuracy of a k-vector coded by k intervals $(\delta_i(e)^-, \delta_i(e)^+)\ \forall i \in [1, k]$ will be $n/2k$ for a space of $2k$ values $(\delta_i(e)^-, \delta_i(e)^+)$.

While we were working on k-vector, in parallel, Claire Diehl in [Diehl 1992] was working on a set of approximations of the causality relation by considering a decomposition of the events set. And in the same way, she showed that Lamport clocks is a decomposition of the events set itself and Mattern vector time a decomposition of the events set in n groups of processes. The coding of the partial order resulting from this decomposition is an exact coding of the causality relation.

Hence, we could think that by refining the decomposition, then augmenting the size of the timestamps, we should observe that the order induced by such timestamps characterizes more precisely the causality relation.

This idea is erroneous in theory, result shown and proved by Dilworth's theorem that we reproduce hereafter [Dilworth 1950].

Theorem 2 *An order of width k can be decomposed in k disjoint chains*

$$E = \uplus_{1 \leq i \leq k} E_i,\ E_i \ chain$$

where the order *width* is the cardinal of one of its largest antichains (subset of unrelated elements).

By consequence, it is illusive to think obtaining better results concerning accuracy of distributed timestamps while realizing arbitrary decompositions of the events set. On the contrary, decompositions which are as closest as possible of a chain decomposition will give a better accuracy when characterizing causality.

6 Conclusion

We studied in this document three main timestamping mechanisms in order to compare their efficiency concerning the identification of causality/concurrency relationships, which we called *accuracy*.

The vector time mechanism is known to fully characterize causality and concurrency as also to reflect exactly the number of preceding event occurrences. It is however space consuming and the two other studied mechanisms are less expensive but less accurate too. Our main result is to quantify, to measure, the amount of imprecision of Jard's and Lamport's clocks. Two different approaches were used to determine this accuracy, which have however the same complexity : the first one rests on numbering the relations which will not be identified correctly by the timestamping mechanisms, the second approach rests on a sub-differential calculus of the accuracy measure and gives the same results. For that purpose, a generic timestamp has been introduced, with a reduced utility as expressed in terms of vector time, but useful to determine the timestamps accuracy. It leaded to show that there exists a family of timestamps that we had called *k-vectors*, whose accuracy however depends on the way subsets of events are arranged.

It is desirable to find an adequate decomposition of the event set, distinct from the one used by Mattern which is a process decomposition. Two processes communicating frequently create chains and grouping these processes should constitute an adequate decomposition of the events set. This is however a difficult problem to solve in the average case, but we can hope to obtain positive results by reducing the framework of this problem to deterministic and synchronous applications.

The purpose of this document was to show that there exists a *compromise* to make, to decide to use logical clocks or interval clocks *according to the characteristics of the execution concerned.* Hence, according to the knowledge one can have of the causality/concurrency measures of the execution, it will be better to use either Jard timestamp or Lamport's one. We showed also that the product *accuracy* $\times$ *space* is constant and equal to n, n being the number of processes.

References

[Charron-Bost 1989] Bernadette Charron-Bost. *Mesures de la Concurrence et du Parallélisme des Calculs Répartis.* PhD thesis, Université Paris VII, Septembre 1989.

[Charron-Bost 1991] Bernadette Charron-Bost. Concerning the size of logical clocks in distributed systems. *Information Processing Letters*, 39(1(11)), July 1991.

[Diehl 1991] Claire Diehl and Claude Jard. Interval approximations of message causality in distributed execution. Rapport Technique 1571, INRIA, Inria-Rennes, Decembre 1991.

[Diehl 1992] Claire Diehl. *Analyse de la relation de causalité dans les exécutions réparties*. PhD thesis. Université de Rennes I. Sept 1992.

[Dilworth 1950] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*. 2(51):161–166. 1950.

[Lamport 1978] Leslie Lamport. Time. clocks and the ordering of events in a distributed system. *Communications of the ACM*. 21(7). July 1978.

[Schwarz 1991] Reinhard Schwarz and Friedemann Mattern. Detecting causal relationships in distributed computations: In search of the holy grail. Internal Report 215/91. Department of Computer Science. University of Kaiserslautern. November 1991.

ISSN 0249-6399